IHSAN DOĞRAMACI BILKENT UNIVERSITY

MECHANICAL ENGINEERING DEPARTMENT

ME 384: Mechatronic Systems

OBSTACLE CLIMBIMG ROBOT

Spring 2016

ATLETİKO MAKİNE

Project Members: Anıl Ünlü
Arda Seçme
Berkay Alp Çakal
Emre Aydemir
Erkan Aksoylu
Verda Saygın

Instructor: Mehmet Selim Hanay

Submission Date: 17 May 2016

Table of Contents

1. Introduction 1

2. Mechanical Design 4

2.1. Initial Design 5

2.2. Final Design 5

3. List of Equipment, Sensors and Components 6

3.1. PIC Microcontroller and Main Platform 5

3.2. Sensors Used 5

3.2.1. Infrared Line Sensor 5

3.2.2. Infrared Sharp Sensor 5

3.3. Servo 5

3.4. Pallet System and the Gears 5

4. Result and Discussion 6

APPENDIX 6

A. APPENDIX A: Coding of the System

1. INTRODUCTION

The main aim of this project was to design and build an automated robot which is capable of following a line and climbing various obstacles with different heights on the line. We were given one base plate two wheels and two DC motors, one IR line follow sensor QTR-8A, one ultrasonic sensor HC-SR04, one PIC16F88, one H-bridge L293D and one Li-Po Battery with 7.4 Volt.

After experimenting the sensors and making preliminary mechanical designs, our systems included two servo motors, two H-bridge, one PIC16F88 microcontroller, one IR line follow sensor and one IR sharp sensor, additionally two pallets with designed and 3D printed gears. The limitations we faced and the design steps will be introduced throughout the report.

This project required combination of informatics, electronics and mechanical design skills.

2. MECHANICAL DESIGN

2.1. Initial Design

Our initial design was to make a system with two DC motors which will compress two springs (or one DC motor which would compress one spring) when the robot is inclined by a servo motor and an rigid arm such that when the springs are reloaded it would make the robot is jumped over the obstacle. The mechanical design and the simple mechanism for the robot to behave when the sensor detect an obstacle was similar; it was designed for the robot to stop when it detects an obstacle, lift itself as the back side and the wheels still in contact with the surface and then the DC motors compresses the springs which will give the robot the intended force to make it jump.

We made several calculations in order to understand how much torque must be delivered from the DC motors to compress the springs, which will store enough energy to make our robot jump. After the calculations we concluded that we will need a high-torque DC motor; which would be both heavy and expensive.

Another limitation was the fact that after the jump, it was highly possible for the robot to land with an unintended deflection; which would make finding the line harder and time consuming.

Due to these limitations and unknown problems which springs may cause we have decided to design a system which climbs the obstacle, not jumps over it.

2.2. Final Design

As mentioned in the introduction our final design includes two servo motors, two DC motors and wheels and two pallets for the climbing mechanism. The main concept of the final design was to make the robot to stop when the sensor detects the obstacle, lift itself by a servo motor and a rigid bar which is implemented under and in the middle of the base plate, as the back wheels are still in contact with the surface. Then make the robot move forward since the front wheels or the pallets make contact with the obstacle; such that when robot tries to climb the obstacle, another servo motor with a rigid arm which is implemented on the back of the baseplate would lift the robot to aid to climb the obstacle.

This design was a safe choice in terms of defining the place that robot lands after climbing, such that we thought we could control the mechanical system with the software easily by defining the pulse width modulation or the angles of the servo motors.

The final design includes two pallets with four wheels and two DC motors. The system is rear-wheel drive, only the back wheels are connected to DC motors and the motors are connected with pallets. We have used the two one piece pallets which have width of the wheels. We have used the rims of the wheels that were given to us and designed gears which would fit the pitch of the pallets. Then we have 3D printed the designed gears which we implemented on the rims afterwards.

We have made two preliminary gear designs and then concluded with a final design which all are presented in the following figures. First two design have not fitted perfectly to the pitches of the pallet which resists the movement of the pallets. Such that we had to make adjustments on the gear design and find the perfect fit for the pitches.

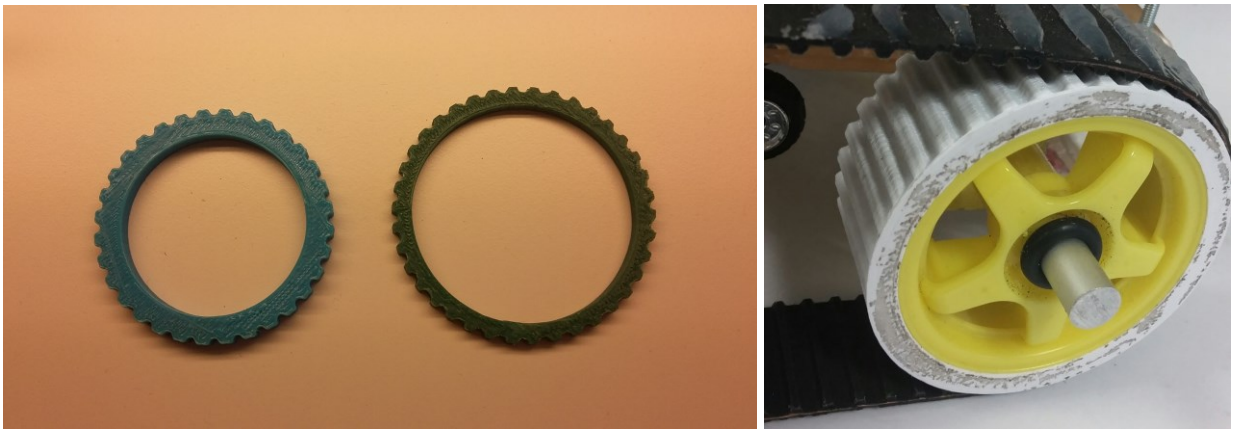

Figure 1-2: On the left first two preliminary 3D printed gears, on the right final 3D printed gear.

For the final design of the main platform, we have added additional two pieces of baseplate in desired dimensions; one small piece of baseplate on the front side of the main platform in order to implement the IR line follow sensor and to protect the IR sharp sensor and another piece of baseplate on the main platform on top of the battery in order to place the breadboard such that it is safe and stationary and the cables can be arranged accordingly.

The servo motor and its rigid arm which was implemented under the main platform caused several problems for the robot to move forward when it is lifted due to friction between the wooden arm and the surface. For the robot to move accordingly with minor effects due to friction and without

any skidding we have added a mini shaft and two wheels which we have ripped from a toy car to wooden arm. Such that in the final design, the robot is lifted with support of two back wheels and the wooden arm which is supported with the mini wheels in the middle.

The outside of the pallets were smooth without any knurl. In order for the pallets to clutch the surface better and climb the obstacles easily, we have made knurls with parallel with each other on the pallets with cold silicon.

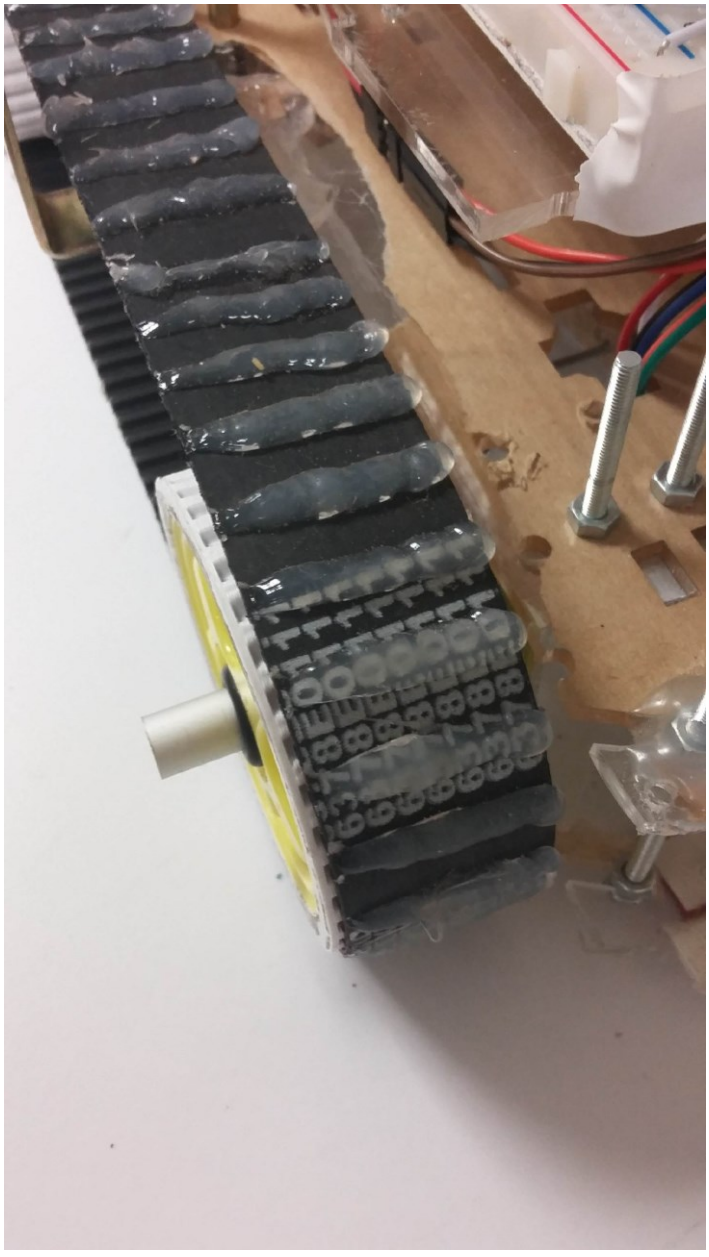

Figure 3: The pallet used in the robot with the silicon knurls added

3. LIST OF EQUIPMENT, SENSORS AND COMPONENTS

3.1. PIC Microcontroller and the Main Platform

We have used single microcontroller PIC16F88 which controls all of our components in the robot. The main platform is the baseplate that we have given. As introduced in the final design we have added two small flexiglass pieces; one to support the breadboard and make it stationary and one to implement the IR line follow sensor. We have decided to use breadboard in our final design.

In addition to the microcontroller we have used 2 L293D, 3 regulators in parallel LM7805 in the main circuitry and one Li-Po battery with 7.4 Volt which was implemented between the breadboard and the main baseplate.

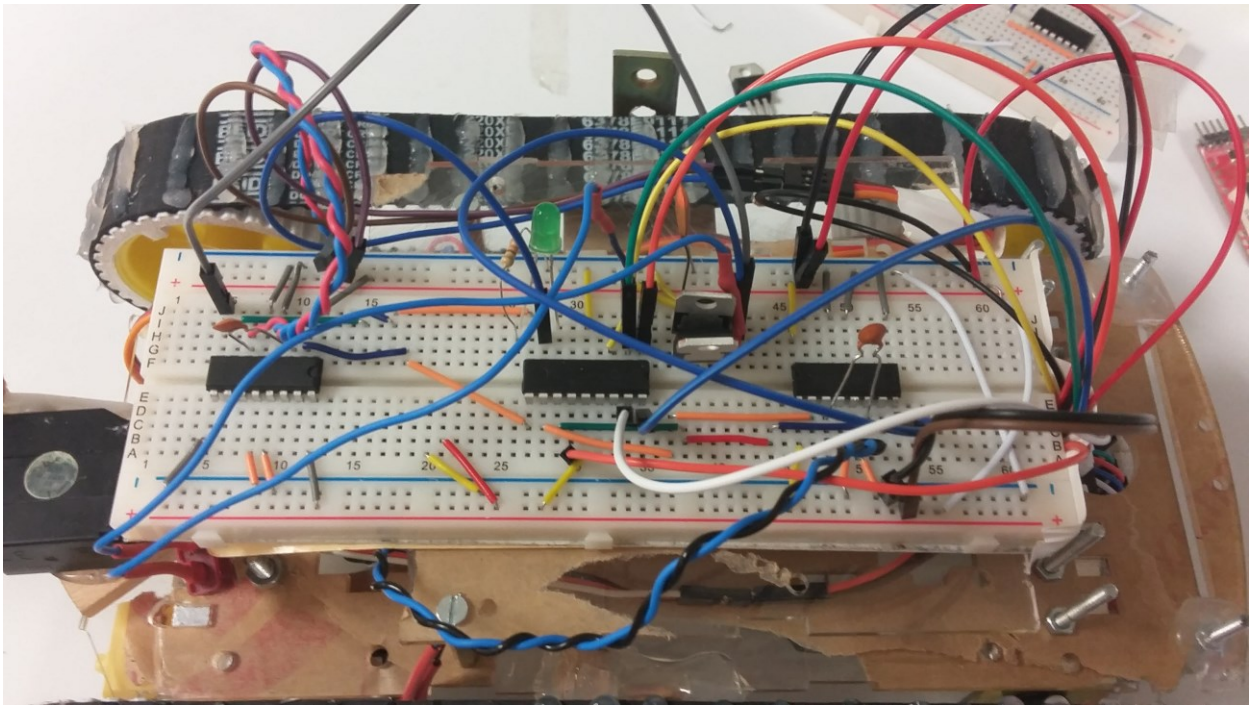

Figure 4: The main platform of the robot and the hardware of the robot

3.2. Sensors Used

In the beginning of the competition we were given one IR line following sensor QTR-8A and one ultrasonic sensor HC-SR04 for detecting obstacles. After experimenting the ultrasonic sensor we have decided it was not feasible and reliable, then chose to use infrared sharp sensor to detect obstacle.

3.2.1. Infrared Line Follower Sensor

We have implemented given IR sensor QTR-8A with a 3 mm height from the surface since its suggested operation range was between 6 mm and 3 mm. We have adjusted our code accordingly as it is implemented with a 3 mm height. It is placed in the front of the main platform which is protected by foam in case the robot crushes an obstacle or lands heavily on the surface

Figure 4: Pololu, QTR-8A Infrared Sensor

3.2.2. Infrared Sharp Sensor

For the obstacle detection we have chosen to use IR sharp sensor which is implemented in the front under the main base plate which is again protected from possible crushes by the main platform itself.

Figure 5: Pololu, 4-30 cm Infrared Sharp Sensor

3.3. Servo Motors

We have used two different servo motors; one under and the middle of the main base plate, the second at the end of the main base plate. Both have wooden arms connected to them which are used to lift the robot for the climbing mechanism. The servo in the middle of the plate is TowerPro MG995, the one at the back of the plate is SM-S3317M. MG995 is more powerful than the other servo motor, which is why we have implemented in the middle to carry the most of the weight of the robot when lifting and climbing the obstacle.

The servo motor and its arm which is in the middle of the baseplate works when the sharp sensor detects an obstacle and the robot stops. The second servo at the end of the plate works after the front wheels climbs the obstacle such that the robot is supported from back and lifted.

4. RESULTS AND CONCLUSION

While making this project theoretical accumulation that learnt throughout three years of mechanical engineering faculty, are made actual. We have used almost all of our mechanical, electronic and software information. Project seems to very hard for us because of we have just one semester, on the other hand we have to finish two projects. We have worked to finish our projects with good teamwork. Throughout the project, we are confronted with many problems and we learnt how to solve these problems and how to take a step that does not pose a problem. And we learnt, how to use hand tools, how to decide which material that should be used. How to make connections between the breadboards and used materials, how to find required components, how to utilize from the materials that we daily use, and lastly we learnt how to solve all of these problems with minimum budget.

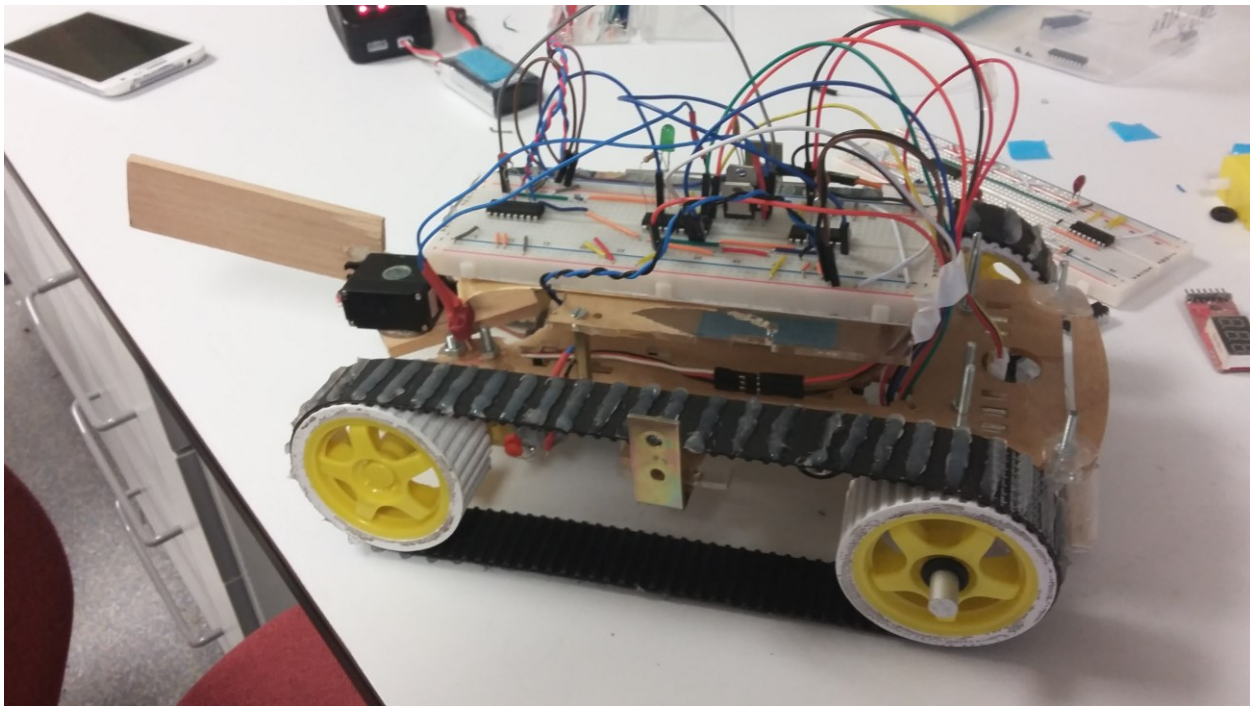

Figure 6: The final state of our robot

The project helped us to increase their familiarity with the sensor, PIC, H-bridge, regulator, servo motors. That both electronic and mechanical design are included in the project offered us the opportunity of learning how to build a robot by making from mechanical components, electronic components to programming themselves. There were the hardware and software parts of the project and these parts are completed by making the mechanical and electronic design and also using the PIC microcontroller to program. Some limitations were included while making the electronic and mechanical design of the robot and these are taken into account while constructing the robot. In addition to these, the project helped us to find the errors in the circuit themselves since there is no indicator where the error is and these situations increase the ability of problem-solving. Mainly, in this project, we were able to find many opportunities to improve ourselves in terms of electronics, mechanics, design and programming.

APPENDIX

APPENDIX A: CODING OF THE SYSTEM

```
#include <xc.h>
#include <math.h>
#define _XTAL_FREQ 4000000


// Declaration Part of Methods
void linefollower(void); //LineFollower Read
void linefollowerwork(void); //LineFollower Work
void turnleft(void);
void turnright(void);
void goforward(int forwardspeed);
void goback(void);
void gobackinterrupt(void);
void checkdistance(void);
void stop_motor (void);
void servorotate30 (void);
void servorotate0 (void);
void passing(void);
void back_servo30(void);
void back_servo0(void);

// Declaration Part of Variables

unsigned short linefollowsensor1;
unsigned short linefollowsensor2;
unsigned short linefollowsensor3;
unsigned short linefollowsensor4;
unsigned short linefollowsensor5;
unsigned short linefollowsensor6;
unsigned short ADCResult;



// Main Function

void main(void)
{
      OSCCON=0b11101000;              // Crystall Freq.
      OPTION_REGbits.nRBPU=0b0; // Pull-Ups Enable Command via
OPTION_REG
      ANSEL=0b00011111;   // Analog Selection
```

```
	TRISA=0b00011111;  // I/O Determination of TRISA
	TRISB=0b00000000;  // I/O Determination of TRISB
	PORTA=0;
	PORTB=0;
	ei();		// Enable Interrupt
	INTCON=0b10010000;  // Now, we set RB0 for interrupt and
clear interrupt conditions on FLAGS

	//servorotate30();
	__delay_ms(1000);
	__delay_ms(1000);

	RB1 = 0;

	//passing(); goforward(20);	goforward(20);
	goforward(20);


	while(1){
		// settlement time
		linefollower();
		linefollowerwork();
		checkdistance();
		//goforward(20);
		//servorotate0();
		//back_servo30();
	__delay_ms(1000);__delay_ms(1000);
		//back_servo0();	__delay_ms(1000);
	__delay_ms(1000);


		while (ADCResult > 200 && ADCResult < 400)
{goforward(10);	checkdistance();}


		while (ADCResult > 400)
		{
			passing();
			//stop_motor();
			checkdistance();
		}


	} // end of while loop

}  // end of main
```

```
void passing(void) {

      RB1 = 1; stop_motor(); __delay_ms(1000);

      servorotate0();
      servorotate0();
      servorotate0();

      servorotate0();
      servorotate0();
      servorotate0();


      for (int i=0; i<6; i++){
            goforward(19);
            servorotate0();
      }

      servorotate30();
      servorotate30();
      servorotate30();

      for (int i=0; i<4; i++){
            goforward(20);
            back_servo30();
      }

      back_servo0();
      RB1 = 0;
}


// Line Following Code - Analyze of Line Position

void linefollower(void){

                        // Selection of Maximum Device
Operating Frequency
                        // Tad vs. Maximum Device Operating
Frequencies Table must be
                        // seen for PIC 16F88
                        // Max. Device Frequency is set as 5
MHz
```

```
                        ADCON0bits.ADCS0 = 0b1; //Clk division
factor = 8 TOSC
                        ADCON0bits.ADCS1 = 0b0; //Clk division
factor = 8 TOSC
                        ADCON1bits.ADCS2 = 0b0; //Clk division
factor = 8 TOSC


/////////////// linefollowsensor1 ------ sensor 1 ////////////

                        ADCON1bits.ADFM = 0b1; // The result is
right-justified.
                        ADCON0bits.ADON = 1; // A/D converter
module is operating
                        // A/D Channel Selection
                        // A/D Channel is selected as AN0 for
PIC16F88
                        ADCON0bits.CHS = 0b000;
                        __delay_us(25); //Waits fr the
settlement.
                        ADCON0bits.GO = 1; //Starts ADC
conversion. SEE ADCON0 register.


                        while (ADCON0bits.nDONE) continue;
//wait till ADC conversion is over
                        linefollowsensor1 = (ADRESH<<8) +
ADRESL ; //Merging the MSB and LSB


/////////////// linefollowsensor2 ------ sensor 2 ////////////

                        ADCON1bits.ADFM = 0b1; // The result is
right-justified.
                        ADCON0bits.ADON = 1; // A/D converter
module is operating
                        // A/D Channel Selection
                        // A/D Channel is selected as AN1 for
PIC16F88
                        ADCON0bits.CHS = 0b001;
                        __delay_us(25); //Waits fr the
settlement.
                        ADCON0bits.GO = 1; //Starts ADC
conversion. SEE ADCON0 register.

                        while (ADCON0bits.nDONE) continue;
//wait till ADC conversion is over
```

```
                        linefollowsensor2 = (ADRESH<<8) +
ADRESL ; //Merging the MSB and LSB

// //////// linefollowsensor3 ------ sensor 3 ////////////

                        ADCON1bits.ADFM = 0b1; // The result is
right-justified.
                        ADCON0bits.ADON = 1; // A/D converter
module is operating
                        // A/D Channel Selection
                        // A/D Channel is selected as AN2 for
PIC16F88
                        ADCON0bits.CHS = 0b010;
                        __delay_us(25); //Waits fr the
settlement.
                        ADCON0bits.GO = 1; //Starts ADC
conversion. SEE ADCON0 register.

                        while (ADCON0bits.nDONE) continue;
//wait till ADC conversion is over
                        linefollowsensor3 = (ADRESH<<8) +
ADRESL ; //Merging the MSB and LSB

// ////////////// linefollowsensor4 ------ sensor 4
////////////

                        ADCON1bits.ADFM = 0b1; // The result is
right-justified.
                        ADCON0bits.ADON = 1; // A/D converter
module is operating
                        // A/D Channel Selection
                        // A/D Channel is selected as AN3 for
PIC16F88
                        ADCON0bits.CHS = 0b011;
                        __delay_us(25); //Waits fr the
settlement.
                        ADCON0bits.GO = 1; //Starts ADC
conversion. SEE ADCON0 register.

                        while (ADCON0bits.nDONE) continue;
//wait till ADC conversion is over
                        linefollowsensor4 = (ADRESH<<8) +
ADRESL ; //Merging the MSB and LSB
}
```

```
// Interrupt Service Routine

void interrupt InfraredSensor(void){

	if(INTCONbits.INTF==1)    //if the interrupt actually
happened on B0
	{
		gobackinterrupt();
		INTCONbits.INTF=0; //Clear the interrupt condition
	}
}


void goforward(int forwardspeed)
{
	int offtime=20-forwardspeed;
	RB4=0;
	RB5=0;
	RB3=1;
	RB6=1;
	for (int i=0;i<forwardspeed;i++)  __delay_ms(5);

	if(offtime>0)
		{
				RB4=0;
				RB5=0;
				RB3=0;
				RB6=0;
			for (int i=0; i<offtime;i++) __delay_ms(5);
		}

}


void goback(void)
{
	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	int backspeed=10;
	int offtime=20-backspeed;
	RB4=1;
	RB5=1;
	RB3=0;
	RB6=0;
	for (int i=0;i<backspeed;i++)  __delay_ms(5);
```

```
	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	for (int i=0; i<offtime;i++) __delay_ms(5);

}

void gobackinterrupt(void)
{
	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	int backspeed=4;
	int offtime=20-backspeed;
	RB4=1;
	RB5=1;
	RB3=0;
	RB6=0;
	for (int i=0;i<backspeed;i++)  __delay_ms(50);

	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	for (int i=0; i<offtime;i++) __delay_ms(50);

}

void turnleft(void)
{
	RB5=0;
	RB6=0;
	RB3=1;
	RB4=0;
	for (int i=0;i<20;i++)  __delay_ms(5);

	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	for (int i=0; i<0;i++) __delay_ms(5);

}
```

```
void turnright(void)
{
	RB5=0;
	RB6=1;
	RB3=0;
	RB4=0;
	for (int i=0;i<20;i++)  __delay_ms(5);

	RB4=0;
	RB5=0;
	RB3=0;
	RB6=0;
	for (int i=0; i<0;i++) __delay_ms(5);
}

void linefollowerwork(void)
{
	//while(linefollowsensor1<512 && linefollowsensor2<512 &&
linefollowsensor3<512 && linefollowsensor4<512){
			//goback();
			// follower();     }

	while(linefollowsensor1>512 && linefollowsensor2<512 &&
linefollowsensor3<512 && linefollowsensor4<512)
		{
			turnleft();
			linefollower();
		}
	while(linefollowsensor1<512 && linefollowsensor2<512 &&
linefollowsensor3<512 && linefollowsensor4>512)
		{
			turnright();
			linefollower();
		}
	goforward(15);
}

void checkdistance (void)
{
	ADCON1 = 0b10000000; //right justified

	//ADCON0 construct
	ADCON0bits.ADCS  = 0b01;   //Selecting the clk division
factor = FOSC/8
	ADCON1bits.ADCS2 = 0b0;   //Selecting the clk division
factor = FOSC/8
```

```
      ADCON0bits.ADON = 1; //start to operate the ADC circutiry
      ADCON0bits.CHS = 0b100; //select analog channel an4

       __delay_us(50); //Waits for the acquisition to complete
    ADCON0bits.GO = 1;   //Starts ADC conversion
    while (ADCON0bits.nDONE) continue;   //wait till ADC
conversion is over

    ADCResult = (ADRESH<<8) + ADRESL ;   //Merging the MSB and
LSB

}

void stop_motor (void)
{
      RB4=0;
      RB5=0;
      RB3=0;
      RB6=0;
      __delay_ms(100);
}


void servorotate30(void)
 {
             for(int i=0;i<20;i++)  //send the position
information twenty times
             {
                   RB2=1;
                   __delay_ms(1.4);
                   RB2=0;
                   __delay_ms(18.6);
             }
}
void servorotate0(void)
 {
             for(int i=0;i<20;i++)  //send the position
information twenty times
             {
                   RB2=1;
                   __delay_ms(0.4);
                   RB2=0;
                   __delay_ms(19.6);
             }
}

void back_servo30(void)
```

```
 {
             for(int i=0;i<20;i++)  //send the position
information twenty times
             {
                    RB7=1;
                    __delay_ms(1.3);
                    RB7=0;
                    __delay_ms(18.7);
             }
}
void back_servo0(void)
 {
             for(int i=0;i<20;i++)  //send the position
information twenty times
             {
                    RB7=1;
                    __delay_ms(0.4);
                    RB7=0;
                    __delay_ms(19.6);
             }
}
```